SIG SPATIAL CUP Report: Constrained Map Generalization

Siva Ravada[1] Xin Chen[2] Zhi Liu[3] Xi Zhang[4]
[1]Oracle, USA (siva.ravada@oracle.com)
[2]HERE, A Nokia Company (xin.5.chen@here.com)
[3]University of North Texas (ZhiLiu@my.unt.edu)
[4]Illinois Institute of Technology (xzhang22@hawk.iit.edu)

ABSTRACT
The 22*nd* ACM SIGSPATIAL Conference on Advances in Geographic Information Systems (GIS) was held in November of 2014 in Dallas, Texas, US. Following the success of last two events, we organized the 3rd programming contest associated with the conference, called the SIGSPATIAL GIS 2014. The subject of the competition was constrained generalization, which aims to generalize geometric features in the context of topological constraints. We describe the contest details, and the results, as well as the lessons learned during the process.

Categories and Subject Descriptors
H.2.8 [Database Applications]: Spatial databases and GIS; D.2.8 [Metrics]:

General Terms
Algorithms, Performance, Theory, Map Generalization

Keywords
Location based services, Map Generalization, Cartography

1. INTRODUCTION
ACM SIGSPATIAL [1] addresses issues related to the acquisition, management, and processing of spatially-related information with a focus on algorithmic, geometric, and visual considerations. The scope includes, but is not limited to, geographic information systems (GIS). ACM SIGSPATIAL [1] is the annual conference sponsored by SIGSPATIAL group. Along with the conference, SIGSPATIAL GIS Cup, which is a algorithmic programming contest focusing on GIS related problem, is also held since 2012 (SIGSPATIAL GIS CUP 2012 [2], 2013 [3]). SIGSPATIAL GIS CUP 2014, similar to the above competitions is open to the entire undergraduate and graduate student population over the world.

Based on the suggestions we got in the last couple of years, the competition was launched by the end of January and ended in July, which gave the competitors more time to work on the problem. The authors of this paper represent the organizing committee members of this year's competition. This paper discusses the SIGSPATIAL GIS Cup 2014, serving as a record of the contest rules, data, winners, and the lessons we learned during the process.

SIGSPATIAL '14, November 04 - 07 2014, Dallas/Fort Worth, TX, USA

http://dx.doi.org/10.1145/2666310.2666419

2. Constrained Generalization
Geometry generalization is a well known concept in the field of cartography (http://en.wikipedia.org/wiki/Cartographic_generalization). This concept is used in producing maps with less detail according to map scale. The algorithms used for geometry generalization usually concentrate on the techniques for simplifying individual geometry objects. When these algorithms are applied to a map, the result might not be what the user expects.

Consider a map that displays a set of state boundaries at a very detailed level. Some states share boundaries with other states and the areas of the states do not overlap with each other. In addition, let's add a set of cities to the map. Some of these cities might be very close to the state boundaries. Now let us consider a simplified version of the same map. If geometry generalization algorithms are applied to each state boundary independently, the resulting map might not be accurate. For example, the simplified state boundaries might not align with each other exactly as they did before the simplification. Some of the cities that are very close to the state boundaries might now be in a different state if the simplified state boundary now falls on the other side of the point used to represent the city.

To facilitate this type of map simplification, it is often desirable to break the state boundaries into different line geometries so that all shared boundaries are represented as unique line geometries. These lines are then simplified and connected back together to form the state boundaries. With this approach, the state boundaries will still preserve the non-overlapping property they had before the boundary is simplified. But this by itself does not guarantee that the cities still maintain their relative position with respect to their state boundaries.

This year's SIG Spatial competition explores this map generalization problem and challenges the students to come up with novel solutions for this very useful problem. The scope of the problem is limited to allow the students to develop a meaningful solution in a short amount of time.

Problem Definition
Input: A set of linear geometries that bound polygonal regions and a set of constraining points.

Objective: Simplify the linear geometries such that the relationship between the constraining points and linear geometries before and after the simplification does not change. In addition, the topological relationships between the original set of input linear geometries does not change after the simplification.

Description: Geometry generalization is a well known area and there are two main algorithms to solve this problem: (i) Douglas-Peucker [4] and (ii) Visvalingam-Whyatt [5]. These techniques can be applied to linear and polygonal geometries to produce generalized geometries. The same techniques can also be applied to produce generalized maps. In map generalization one of the difficult problems is to maintain the topological consistency between adjacent polygonal regions during the simplification process. This is explained with the help of the following figures.

Consider a set of 6 lines: L1 .. L6 arranged as in Figure 1.

Figure 2 shows a valid simplification of these 6 lines. Note that the end points of each of the lines do not move. Only some of the intermediate points of the lines are deleted. And Figure 3 shows an invalid simplification of these 6 lines since lines L1 and L2 cross each other after simplification.

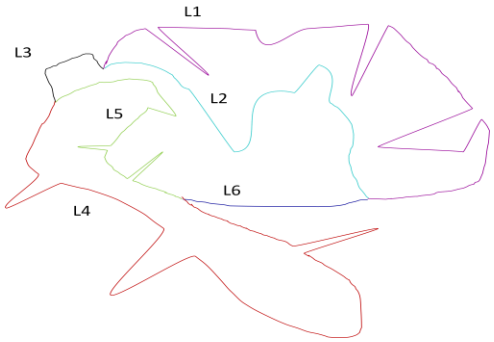

Figure 1. Input data with 6 lines.

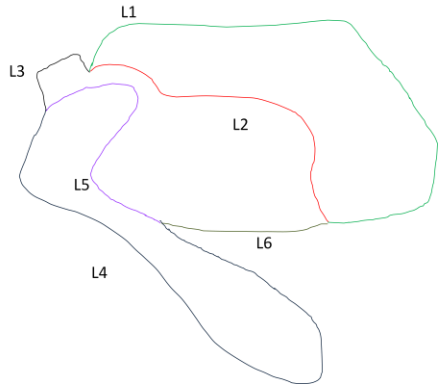

Figure 2. A valid simplification of the 6 lines from Figure 1.

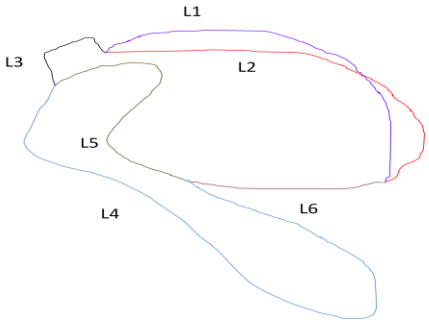

Figure 3. An invalid simplification of lines from Figure 1.

Next few figures show example data set with control points. Figure 4 shows input of 6 lines with 5 control points. Figure 5 shows a valid simplification of these lines while Figure 6 shows an invalid simplification of these lines.

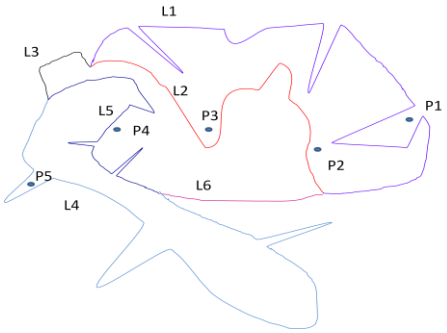

Figure 4. Input data with 6 lines and 5 control points.

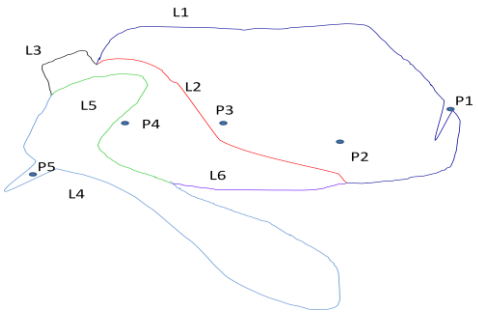

Figure 5. A valid simplification of input from Figure 4.

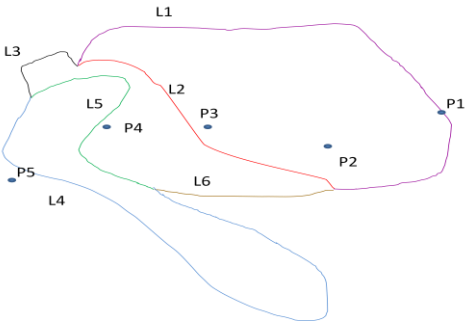

Figure 6. An invalid simplification of input from Figure 4.

Note that in Figure 5, P1 forces line L1 not to be simplified further and P5 forces L4 not to be simplified further. The

simplification in Figure 6 is invalid as points P1 and P5 change their relative position with respect to L1 and L4.

Assumptions about the data and simplification process

1. Data is assumed to be in Cartesian space.
2. Linear geometries do not have self-intersections.
3. Linear geometries do not intersect with other linear geometries except at end points.
4. The constraining points do not intersect any of the linear geometries.
5. Number of lines will vary for each test.
6. Number of points will vary for each test.
7. The line generalization algorithm is expected to delete some of the input points to generalize the lines. It is not expected to add vertices that are not part of the input.
8. The number of vertices in the generalized line are less than or equal to the number of vertices in the original line geometry.
9. The line generalization algorithm should not change the end points of the input line geometries. That is, it can remove any intermediate vertices except the end points. This preserves the original end point coincidences of the input line geometries.

Evaluation Rules

1. The time to process all the input data will be the main criterion for the evaluation of the solution. So the solution with the best rate (points reduced per minute) is the winner.
2. Each line will be assigned a score that is equal to the number of points reduced compared to the original geometry for that line.
3. The accuracy of the result is also considered. If a line is simplified, but it violates the topological constraints of the input, the simplification for that line will not be considered. That is, even if x number of vertices are deleted from that line; the score for that line will be zero.
4. The program should be able to take a parameter that specifies the number of points to be removed from the input.
5. We will evaluate the input program with different values for this parameter. Final score will be averaged over all the different runs.

Input Data Format

The first section of the input will describe the input line geometries in GML 2.1.1 format. The format will be an ID, followed by the GML for the line. The second section of the input will describe the input point geometries in GML 2.1.1 format. The format will be an ID, followed by the GML for the point.

Output Data Format

Output should be the set of simplified line geometries along with their IDs. The point geometries do not appear in the output.

3. Submission and Evaluation

3.1 Submission

Submission is managed by the online conference management via the URL https://cmt.research.microsoft.com/GISCUP2014. The deadline for the submission was August 1st, 2014. Participants were required to submit a single zip file that that contains the original source code and any dependencies, a readme.txt file for any special instructions on how to compile the submitted code, and a single executable file name **Simplify.exe**. Submission of the source code was mandatory to ensure originality of the submitted work. The Simplyfy.exe accepts three command line parameters. The usage of the Simplify.exe program is as follows:

Simplify <PointToRemove> <LineInputFilePath> <PointInputFilePath> <OutputFilePath>

1. <PointsToRemove>: Specifiies the minimum number of vertices that must be removed from the input line segments. The program can remove more than this minimum specified number, but it has to remove at least this minimum number of vertices.
2. <LineInputFilePath>: Specifies the line data file. Each line data file is a single test case and contains a set of line geometries. The example format of the files can be found at Problem Definition section.
3. <PointInputFilePath>: Specifies the point data file. Each point data file is a single test case and contains a set of points. The example format of the files can be found at Problem Definition section.
4. <OutputFilePath>: Specifies the result file where the program is expected to store the output. The example format of the output files can be also found at Problem Definition section.

3.2 Evaluation Metric

Evaluation of the submission is done using the following algorithm. First each simplified line is checked to make sure it does not violate any constraints. If it violates any constraints, we reject the simplification of that line and assume that is not simplified. After this check is done, we count the number of vertices removed after simplification and assign this as the score. We then divide this score with the actual execution time to find a grade. This grade is used in the final ranking of the submissions.

Using this method, we are able to assess the submissions for correctness and efficiency of the simplification process. All submissions are evaluated on Microsoft Windows System (Intel Quad-Core, 64bit).

3.3 Evaluation Process

In addition to the three training data sets provided to the participants, two more data sets are used to evaluate each submission. Each submission is executed 12 times and the timings for the last 10 runs are recorded. The results for each data set are then checked for correctness and assigned a score. Scores for all

Title	Authors	Time (ms)	Score	Grade
An Efficient Method of Map Generalization Using Topology Partitioning and Constraints Recognition	Hongtai Zhang, Jian Dai, Kuien Liu, Huidan Liu, Danhuai Guo	139	57491	413.604317
A Fast Algorithm of Geometry Generalization	Yuwei Wang, Danhuai Guo, Kuien Liu, Yan Xiong	159	57184	359.647799
Greedy Map Generalization by Iterative Point Removal	Yanzhe Chen, Rong Chen, Haibo Chen, Binyu Zang, Yin Wang	211	58221	187.205788

the 5 data sets are added to generate a total score. Similarly, the timings for each data set are added up to generate a total time of execution for each submission. These two numbers are then used to compute the grade.

Some teams have submitted multiple programs to the competition. We only took the best submission from each team for the final rankings to make sure we get 3 different winning teams. In addition to the top 3, we also selected the 4th placed team as well to participate in the conference as the final grade between the 3rd and 4th place teams is very small. And finally we also selected a 5th submission as this was the only submission to simplify all the input data sets without any constraint violations.

Overall, we received 34 submissions. Out of which some had issues running the programs on the test machine. We gave the participants with problematic submissions 1 week grace period to fix the issues and resubmit the programs. At the end of that 1 week period we had 32 valid submissions. Of these, only 26 submissions were successful in completing the simplification on the two new data sets. So in the end only 26 valid submissions were considered for the final rankings.

The number of points that were required to be removed for each data set is as follows.

1. Set 1: Remove 500
2. Set 2: Remove 500
3. Set 3: Remove 5000
4. Set 4: Remove 23000
5. Set 5: Remove 22000

4. RESULTS

Based on the final grades, we selected the top three teams and two additional submissions for participation at the main conference. Table 1 summarizes the final results for the top three submissions.

Most of the submissions had trouble simplifying lines that are looping lines: that is lines that have the same start and end point. And when this line is simplified, there should be at least 4 vertices left in the line to keep it as a valid line segment. Other than this case, different submissions failed to simplify different types of lines geometries. But there is no common thread among these failures.

5. LESSONS LEARNED

We learned several lesons during the competition and the evaluation process and we would like to share these with the organizers of the future SIGSPATIAL GIS CUP competitions. As SIGSPATIAL GIS CUP requires the participants to submit executables, rather than the result files (as the other competition have done), we faced several challenges during the evaluation phase: (1) some of the submitted programs lack dependent files/libraries and (2) some of the participants made assumptions based on the training data sets that are not valid for the final data sets used for evaluation. For example, the training data sets have numbers in the decimal format and some numbers in the final two data sets have scientific format. Some submissions failed to read this scientific notation for numbers and crashed during evaluation.

For the first problem, there is no easy solution. May be next year the organizers should think about providing a VM that the participants can use to make sure their submissions work on that VM. Then the organizers can provide a physical machine with the same software as the VM to evaluate the final submissions.

6. ACKNOWLEDGMENTS

We would like to thank Microsoft and NVIDIA for their generous support to sponsor the prizes for the winners of this competition. We also thank the members of the organizing committee of the SIG SPATIAL conference for their support and help during this process.

7. REFERENCES

[1] ACM SIGSPATIAL http://www.sigspatial.org

[2] ACM SIGSPATIAL GIS CUP 2012 http://depts.washington.edu/giscup/home

[3] ACM SIGSPATIAL GIS CUP 2013 http://dmlab.cs.umn.edu/GISCUP2013

[4] David Douglas & Thomas Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", The Canadian Cartographer 10(2), 112–122 (1973)

[5] The Douglas-Peucker algorithm for line simplification: Re-evaluation through visualization M. Visvalingam, J. D. Whyatt Computer Graphics Forum, 9(3), September 1990, pp. 213-228.